Performance Evaluation of Web Sites Using HTTP/1.1 and HTTP/2

Michał Druzgała[1] and Ziemowit Nowak[2(✉)]

[1] brightONE Sp. z o.o.,
ul. Piotra Skargi 3, 50-082 Wrocław, Poland
michal.druzgala@brightone.pl
[2] Faculty of Computer Science and Management,
Wrocław University of Science and Technology,
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
ziemowit.nowak@pwr.edu.pl

Abstract. Introduction into the problem matter of the chapter. Overview of similar work (literature). Presentation of techniques of improving the performance of websites used in the HTTP/1.1 protocol and the problems they can generate in the context of the new possibilities of the HTTP/2 protocol. Description of the research stand and experiments. Discussion of the results of measurements of the speed of downloading websites using various techniques. Formulation of conclusions. The directions for further research were pinpointed.

Keywords: Domain sharding · Resource inlining · Concatenation
Minification

1 Introduction

When designing a website, one can influence how the site will be processed on the target user's side through a properly designed structure of HTML, CSS or JavaScript code, as well as by the proper preparation of the resources that make up the site in terms of structure as well as location. As a result, an important aspect of website's design is its performance.

When designing an efficient website, i.e. one that will be quickly processed and displayed on the client's side, it is possible to influence its reception, the general feeling of smoothness and convenience of using the website. A number of techniques, or proven methods are helpful here that in a universal way fit into the concept of shortening the loading times. They are based, inter alia, on the knowledge of the principles of the HTTP protocol, its limitations and disadvantages [1].

Currently, however, the use of some techniques is questionable in view of the increasing share of the new version of the HTTP – HTTP/2 protocol, which offers improvements resulting from the growing demand for efficient websites [2].

P. Gaj et al. (Eds.): CN 2018, CCIS 860, pp. 142–157, 2018.
https://doi.org/10.1007/978-3-319-92459-5_12

Due to the lack of a full understanding of the impact of the techniques used so far on HTTP/2 performance, there appeared a need to analyse the classic rules developed for the HTTP/1.1 protocol, in terms of their further relevance in view of the prospect of a significant increase in the new version of the protocol.

2 Related Works

Liu et al. [3] conducted a study of the effectiveness of the HTTP/2 protocol in terms of displaying a site on a mobile device. Using copies of the 200 most popular sites according to the Alexa rankings on the local server, they conducted a comparative analysis of HTTPS and HTTP/2 protocols for the impact of factors such as Round-Trip Time (RTT), bandwidth, packet loss rate and size of resources. The research showed the advantage of the HTTP/2 protocol, among others in situations when there was a need to download many smaller objects and the low rate of packet loss. The authors also noted the possible undesirable effects associated with the use of good practices derived from the previous version of the protocol, such as domain sharding, concatenation or inlining.

Corbel et al. [4] examined Page Load Time (PLT) using an unencrypted version of the HTTP/1.1 and HTTP/2 protocols for the prepared "average site", consisting of various types of resources stored on several different domains. The measurement scenarios assumed several separate delay profiles and the loss rate of the packet. The analysis showed the advantage of the HTTP/2 protocol in each scenario. The authors stressed the need to extend the research with the new mechanisms available in the HTTP/2 protocol and scenarios taking into account the size of the TCP window at the client's.

de Saxc'e et al. [5] analysed the impact of the HTTP/1.1 protocol change on the HTTP/2 protocol in the context of web site load performance. The 20 most popular sites according to the Alexa rankings were used for the purposes of the research and they were adapted to the needs of the local measurement environment and the environment similar to network realities (a dedicated HTTP server located outside the local network). PLT was determined as a measure of the loading performance of the websites. Measuring scenarios took into account various pre-defined network conditions (including ADSL and 3G). The research consisted of a number of scenarios using the new HTTP/2 protocol: compression, multiplexing, server push and prioritization. As a result of the research, the authors found the advantage of the new version of the protocol. They noted, however, worse performance in 3G network conditions (significant vulnerability to packet loss).

Varvello et al. [6] developed a measurement platform to track the use of the HTTP/2 protocol among the million most popular websites according to the Alexa ranking. Their research showed, inter alia, that the owners of most sites that adopted the new version of the protocol retained techniques developed for the purposes of the HTTP/1.1 protocol. Among those inlining or domain sharding can be mentioned. About 80% of registered sites supporting the new version of the protocol obtained better PLT. The best results were recorded for 3G networks in Europe.

Zarifis et al. [7] performed PLT analysis after switching to the new version of the HTTP protocol. For research, they used data from the Resource Timing API for approximately 280,000 visits by Akamai CDN users. Based on the modelling of the HTTP/2 server behaviour, using the collected visits measurements for Akamai servers (HTTP/1.1), they tried to analyse the theoretical decreases in PLT times. The authors showed that theoretically, the use of new techniques offered by HTTP/2 (server push, prioritization) would have a positive impact on the final loading times of websites.

Stepniak and Nowak [8] examined the impact of the new version of the protocol on web applications of the Single Page type (SPA) using different types of techniques to accelerate the loading of the site (concatenation, compression, minification). In the context of the HTTP/2 protocol use, they also examined the capabilities of the server push technique. The research showed a significant advantage of the HTTP/2 protocol in the event of having to download many smaller resources. The only aspect that had a negative impact on the loading times of the site was the use of the server push mechanism.

Rosen et al. [9] analysed the potential benefits of using the server push technique available for the new version of HTTP from the mobile clients' perspective. Their research involved 50 sites from among the top 500 of the most popular according to Alexa's ranking, adapted to the local conditions. As a result, they found measurable benefits stemming from the use of the new technique. Still they pointed to reduced efficiency of the technique for sites that use multiple servers to store resources, and in the case of a scenario involving a mobile client with a device inferior in performance, which may negatively affect the final processing time of the site (longer download times of the HTML file).

Kim et al. [10] carried out research on the efficiency of loading sites that use multiple domains to store resources in the context of the HTTP/2 protocol. They performed PLT measurements for typical Korean sites (which usually use multiple domains in their structure), copied to the local machine to be analysed. To construct the research environment, they used a server natively supporting the HTTP/2 protocol. The research showed increased PLT in all the adopted scenarios.

Seemakhupt and Piromsopa [11] analyzed scenarios in which the use of stream multiplexing using the HTTP/2 protocol brings the greatest benefits and may lead to a deterioration in site loading efficiency. They adopted two scenarios in the research. In the first one, the client individually queried the HTTP server for the resources of the site. In the other, an additional client was burdening the link. The results showed that the use of the HTTP/2 protocol using multiplexing performs better under low load conditions. However, under load conditions, the new protocol version performs worse than the HTTP/1.1 protocol.

Prokopiuk and Nowak [12] examined the impact of the new version of HTTP on the performance of the site from the perspective of the end user (User-Perceived Web Application Performance). They used WebPagetest tool to measure the performance of the examined websites, installed on the local client machine. The tool allowed them to acquire a number of measurements that are

relevant to the user displaying the site. Network conditions were simulated by the authors with the help of the Dummynet tool. As a result of the research, the authors found a beneficial effect of the server push mechanism in almost every scenario. The greatest benefit came from attaching CSS files and a background image.

3 Review of Issues Related to the Use of Old Techniques for the New Protocol

With the introduction of the new version of the protocol, some of the techniques used for the HTTP/1.1 protocol no longer bring time benefits or – even worse – negatively affect the total loading times of sites. This section presents problems resulting from preserving the old solutions after switching to the new version of the HTTP protocol and suggestions of the new approaches when designing sites using the HTTP/2 protocol [1,2].

3.1 Domain Sharding

With the development of websites, the need for parallel processing of resources increased. In the context of the HTTP/1.1 protocol, this meant using additional TCP connections, even within the same domain. Due to the overhead generated by additional connections, browser developers established internal limits of maximum connections within the domain in order to eliminate the risk of possible overloads of the system. For example, for Google Chrome, the maximum connection limit is 6. In many cases, the connection limits were too low, which motivated the creation of a new technique - domain sharding.

The idea of the technique is simple. In order to "cheat" the browser, some resources are shown in another, additionally created domain, which in fact points to the same server. Thanks to the use of the technique, it is possible to exceed the set limits and increase the parallel processing of resources in a browser-independent manner.

Formulating the Problem. From the perspective of the new mechanism and message exchange structure for the HTTP/2 protocol, technical improvements cease to be improvements. The HTTP/2 protocol natively supports full parallelization and multiplexing of messages within a single TCP connection.

The use of domain sharding for the HTTP/2 protocol generates overhead associated with, among other things, the creation of new TCP connections, while there is no need to do so. It is not possible either to use the full potential of mechanisms for prioritizing and controlling the flow of messages due to the additional TCP connections created. To some extent, the header compression mechanism also suffers, as it must operate in this situation separately for each connection, introducing a certain degree of redundancy.

Suggested Solution. Domain sharding is the first technique whose discontinuation should be strictly considered. It avoids constraints that no longer exist in the new version of the protocol (the problem of new TCP connections and the limited parallelism of resource processing).

Direct cancellation of the technique shall have a positive impact on the site's performance, which after the change will use the minimum number of TCP connections. In addition, it should be taken into account that in order to effectively implement the prioritization and flow control mechanism, transmission of flows within the minimum number of TCP connections is required. An important advantage is also the correct operation of the HTTP header compression mechanism (the header context is remembered within a single TCP connection).

3.2 Concatenation of Resources

In 2007, the total processing time of the site was only 10÷20%, which included the time to download the HTML document itself [13]. The remaining time was devoted to the processing of additional resources appearing as references in the HTML structure. Due to the growing number of HTTP requests, the overhead associated with performing an increased number of resource queries became more and more important from the performance perspective.

A technique of concatenation, i.e. joining resources of one kind in order to reduce the number of queries, came to the rescue. Merged resources that have been downloaded must be logically processed to reach their condition from before the connection. For merged images (sprites) CSS styles are created that "cut" the one that is needed on the site. There is no need for additional steps for CSS and JavaScript resources.

Merged resources are readable by the browser in this form. The use of the concatenation technique is able to reduce the loading times of sites by up to 50% [13].

Formulating the Problem. In the context of the HTTP/1.1 protocol, the use of concatenation is beneficial by reducing the problem of simultaneous processing of many smaller resources. The HTTP/2 protocol reveals its full potential in the scenario of high resource granulation. The mechanism of multiplexing streams within a single TCP connection is able to handle single resources in the case of parallel processing.

The use of the concatenation technique in the context of the HTTP/2 protocol creates the problem of a longer wait for a specific resource due to its connection with others. Another issue is cache management. The smallest change in the merged resource forces makes it necessary to reprocess the whole (redundancy of data transfer).

Suggested Solution. The concatenation technique is designed, among other things, to reduce separate requests to the server, which has a positive impact on performance in the context of the HTTP/1.1 protocol. Another effect of applying

the technique is also an increase in compression efficiency for related resources, which is a positive aspect also in the new version of the protocol.

Due to the above, this technique should not be completely abandoned, but should not be overused either. One should look for the "golden mean".

3.3 Resource Inlining

The resource inlining technique allows one to achieve the effect of directly downloading resources along with an HTML document in a single HTTP response. The selected resources are "embedded" (inlined) directly in the HTML document, forcing the transfer of the resource along with the structure of the site as a result.

Embedding a resource may look different depending on the type of resource. In the case of CSS styles or JavaScript scripts, it is enough to place the content directly in the HTML structure between relevant tags. In the case of multimedia, it is possible to attach encoded content in Base64 directly in the HTML tag.

This is another way to reduce the number of queries to the server and in a way to prioritize some resources (the embedded resources naturally receive the highest priority). Increased loading times of the base HTML structure should be borne in mind, however, which delay the start of sending requests for external resources.

Formulating the Problem. The resource inlining technique offers measurable benefits in the context of the HTTP/1.1 protocol due to its contribution to solving the problem of parallel processing of many resources and the reduction of excess RTT times caused by the processing of separate resources.

The different characteristics of the HTTP/2 protocol mean that the approach has a negative impact on the website's loading performance due to a breach of the concept of multiplexing and prioritizing the transfer of resources that must be separately identified in the structure of the HTML document. There is also the problem of data duplication in the case of multiple use of the same resource. As well as the problem with the effective use of the cache control mechanism.

Suggested Solution. The new HTTP/2 mechanism, server push, provides an alternative to resource inlining in terms of query reduction. Even the most naive strategy adopted by the server is better than the direct embedment of the resource, over the downloading of which the client has no control.

3.4 Minification

Another aspect of shortening transmission times is the reduction of the direct content of the transferred resource. In the case of resources containing CSS or JavaScript code, this can be achieved using a technique called minification (and obfuscation).

Minification involves reducing the white characters used in the code to improve its readability in the course of its development. White characters are reduced to a minimum while maintaining the semantic correctness of a given code. As a result, the resource should not differ from its predecessor in terms of functionality. Such a modified resource should be included in the document in order to send a request for an already identified resource (one-off overhead related to the preparation of the resource).

Obfuscation (darkening) goes a step further, modifying to the minimum the names given to all programming entities (variables, functions and others) while maintaining syntactic and semantic correctness with the original. The yield in relation to the version of the modified resource is noticeable [13].

The experiment [13] carried out on the modified and obscured JavaScript resources showed a time gain of $17 \div 31\%$ (depending on the size of the resource).

4 Analysis of the Speed of Websites Operation Using Various Techniques

4.1 Research Assumptions

When analyzing the impact of changing the protocol on the performance of the site, the main measure of performance was the size of PLT [14], the time measured between the start of navigation to the site and the moment indicating the completion of processing all additional resources associated with the examined site.

The user perceived performance aspects were not subject to the study, which means that information about the speed and order of rendering resources by a particular browser was not collected and analysed.

The following profiles of the clients visiting the website were adopted:

- desktop client with Google Chrome 59,
- desktop client with Mozilla Firefox 53 browser,
- mobile client (Android) with Google Chrome 59.

As a source of data on the speed of loading the site, only the target client's site (supported browser) was determined.

Bearing in mind the assumptions regarding the site's performance measurement, the decision was made to use the Resource Timing API. It is a JavaScript programming interface that enables one to obtain precise, high-resolution time information (parts of milliseconds) for all resources that are part of the site. Through this interface, it is possible to access information about all HTTP request components about the resource [15,16].

Jetty 9.4 server was chosen to handle HTTP/2 requests. For the proper functioning of the server, it was required to use an additional ALPN (Application-Layer Protocol Negotiation) library, allowing the negotiation of the protocol used at the application layer.

As a basis for the preparation of websites in the simulation environment, a partially configured Spring Boot demo project with the Jetty server was used

[17]. The project was modified to support multiple ports, implementation of the server push mechanism for explicitly defined resources that were part of the selected site and implementation of the mechanism for capturing and recording measurement results.

In order to examine the performance of sites using each protocol individually, a solution was chosen involving the selection of the protocol version by using appropriately assigned ports:

- port 8080: using the HTTP/1.1 protocol,
- port 8443: using the HTTP/2 protocol.

In both cases, connection encryption was used.

Taking into account the variability of network conditions on a larger scale over time, the decision was made to conduct measurements in an isolated local area network where only one server with the examined site was located at the time of measurement, and one client querying server resources.

Being aware of the fact that it is the first visit that is the key from the perspective of the user visiting the site, the decision was made to focus on analysing only the first visit to the site. It means that all techniques that primarily use the cache mechanism will not work in a simulation environment, so they were not taken into account. In order to simulate the continuous effect of the first visits, it was decided to use developer tools in the browsers, and more specifically, the option to disable the cache.

In an isolated environment, reflecting the reality of conditions in terms of bandwidth and prevailing delays while maintaining these conditions unchanged, a bandwidth control and server-side latency mechanism was adopted. Taking into account the possible discrepancy in the effectiveness of the techniques used in terms of network conditions, the following two network profiles were determined subject to separate measurements:

- DSL profile: 2 Mbit/s bandwidth, 5 ms delay,
- 3G profile: 750 Kbit/s bandwidth, 100 ms delay.

Dummynet [18] was used to simulate the established profiles. It is a tool that allows one to emulate specific network conditions in real time. Originally created for FreeBSD, it is now also available for Linux, OSX and Windows distributions. The tool uses ipfw firewall software. Ipfw allows communicating with Dummynet via the command line and setting traffic restrictions that ultimately are ipfw firewall rules. Installed as a service on a network card in Windows, it is able to capture traffic and subject it to various modifications, including bandwidth and delays.

4.2 Research Stand

In order to meet the research assumptions and to ensure precise research results, a dedicated simulation environment was developed (Fig. 1).

The HTTP server was a PC computer with the following parameters:

– Intel Core i5 processor,
– 8 GB RAM,
– Windows 7 32-bit operating system.

The server software was installed on the computer along with the sites to be examined. On its network card, the Dummynet and ipfw drivers were installed and, as a result, it was possible to properly manipulate network conditions from the level of the desktop console.

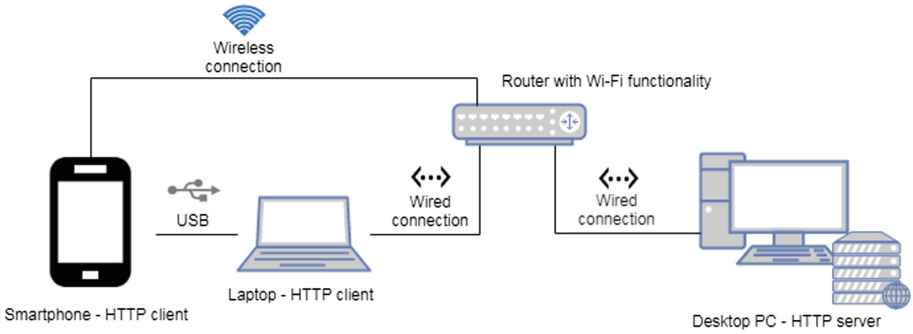

Fig. 1. Schematic drawing of the research stand.

The HTTP client was based on the Dell Latitude E6440 laptop with the following parameters:

– Intel Core i5 processor,
– 8 GB RAM,
– Windows 7 32-bit operating system.

On the HTTP client laptop, Google Chrome and Mozilla Firefox were installed. In both browsers, developer tools and private mode were launched and activated.

The OnePlus One smartphone was used to examine the mobile version of Google Chrome using the following parameters:

– Qualcomm Snapdragon 801 processor,
– 3 GB of RAM,
– Android 6 (Marshmallow) operating system.

During the research, the smartphone-client HTTP connected to the router using Wi-Fi technology in the 802.11n standard.

4.3 Sites Examined

According to the adopted assumption regarding the separation of the applied techniques, in order to carry out measurements for each technique individually, it was necessary to construct separate sites for each technique separately with the division into the "raw" version (without using any technique) and the version using the chosen technique. To minimize the need for additional queries, scripts analysing website load times and any CSS style sheet sources were placed directly in the HTML code.

Concatenation. The main part of the page under investigation for the technique of concatenation were six resources in the form of images in the PNG format. As a raw state, six separate resources defined in the HTML code were determined for the purpose of rendering six images, which involved their complete separation. As a result of such defining the structure of the site, it was necessary to download each resource separately.

After applying the concatenation technique, the resources were merged into one larger resource, which was later logically separated on the client's side into individual resources using the CSS styles attached to the HTML structure.

Due to the concatenation technique, only one graphic resource was referenced, and then its logical "cutting" into individual images took place. As a result, the HTML object code rendered the images by appropriate references to the CSS classes.

Inlining. Six resources in the form of images in PNG format were reused for the purposes of the inlining technique. Also, the "raw" state was determined by the need to download all other additional resources.

Following the application the inlining technique, resources became an integral part of the HTML document. References to additional resources were replaced with references to resources encoded directly in the document using Base64 encoding.

With the "raw" version of the site, a solution utilizing the server push mechanism, available in the new version of the protocol, was also analysed. Due to the architecture of the measurement system, an additional version of the site was created for the purpose of proper site selection on the backend side. On the side of the HTML document, the structure did not change with respect to the "raw" version. Additional graphic resources still had to be downloaded. The difference was due to the use of the server push mechanism on the backend side using the explicit resource selection.

Domain Sharding. A separate site was prepared for the domain sharding research purposes. It contained in its body references to twenty graphics resources that constituted components of a certain graphics composition.

Queries to additional resources were made in a standard way, by including HTML graphics elements. As a result of establishing such structure of the HTML document, it was necessary to query the server for each element separately.

When using the HTTP/1.1 protocol, the limits of simultaneous connections within a single domain had a significant impact on the performance of such a solution. Thus, additional "virtual" domains were used, entered for the purposes of research into the hosts file. Instead of twenty queries for one domain, queries were divided into four additional domains, allocating a maximum of six resources per domain. Behind the additional domains there was exactly the same IP address from which the HTML document was downloaded.

Due to the need to fully use the HTTP/2 protocol, a third version of the site was created, having the same structure, differing only in the port number (8443 instead of 8080), used for queries for additional graphics resources.

Minification. For the purpose of analysing the effectiveness of the minification technique, an appropriate website was also prepared. In the raw document structure there were three references to additional resources in the form of JavaScript libraries and CSS libraries with a significant size in relation to the rest of the site's resources. In the version using the minification technique, three references to additional resources were also used, but this time their internal structure was changed to reduce their size.

4.4 Methodology

Every possible scenario of the user's visit was subject to a separate measurement series. The scenario should be understood as a set of the following factors:

- network profile,
- application of the techniques (or the lack thereof),
- protocol version,
- type of browser (along with the type of HTTP client device).

The measurement series consisted of twenty visits, which were individually recorded in the database as one measurement. Each subsequent visit to the resource page, being an element of a single measurement, proceeded in a sequential manner. Each subsequent measurement was started only at the end of the previous one. Between each individual measurement, a time window of one second was established.

The next step was the data cleansing phase, which consisted of the following activities:

- removal of outliers and measurements with a gross error,
- verification of areas of component values of measurements,
- verification of the correctness of writing to the database.

After cleaning the collected data, each case was analysed in detail. Histograms of purified data were examined to establish the distribution, which directly affected the decision regarding the adopted measurements.

As a result of the analysis, it was decided that the median values from the measurement series would be used as the measure of central tendency for the representation of the results of individual scenarios. The use of median is justified due to the fact that most of the histograms of the measurement series for the determined scenarios had a skewed distribution.

4.5 Results of the Analysis and Conclusions

The lists of average PLT times are presented in four tables. In the case of scenarios that differ from the "raw" scenario for the HTTP/1.1 protocol, the percentage gain (PLT) of the PLT time in relation to this scenario is also given.

Taking into account the current trends in the context of users' preferences regarding web browsers and the continuous increase in the share of mobile clients, after a thorough analysis of the interpreted results, the following conclusions can be drawn regarding the use of selected techniques in conditions that take into account the potential use of HTTP/2 for sites with similar of character in relation to the examined websites.

Concatenation. When deciding on changes in the context of applying the concatenation technique, several factors should be taken into account (Table 1). The trends regarding the website's clients in terms of used browsers and platforms (mobile/desktop) should be examined.

Table 1. Comparison of Page Load Time for **concatenation** scenarios.

Concatenation		HTTP/1.1			HTTP/2.0			
		raw	active		raw		active	
		PLT [ms]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]
DSL	Chrome PC	148.0	152.0	−2.70%	141.0	4.73%	149.0	−0.68%
	Firefox	232.0	184.0	20.69%	192.5	17.03%	181.0	21.98%
	Chrome mobile	272.0	276.0	−1.47%	267.5	1.65%	268.5	1.29%
3G	Chrome PC	453.0	446.5	1.43%	647.0	−42.83%	647.0	−42.83%
	Firefox	1893.0	1278.0	32.49%	1489.0	21.34%	1605.5	15.19%
	Chrome mobile	636.0	613.5	3.54%	757.0	−19.03%	769.5	−20.99%

If the majority of users of the site are desktop users with a constant Internet connection using the Mozilla Firefox browser, it is recommended to continue using the concatenation technique (17.03% gain PLT for the "raw" version and 21.98% gain for the "active" version).

Otherwise, what is meant by a larger share of mobile clients or using the Google Chrome browser, it is discouraged to continue using the technique due to

better PLT results without using the technique in the context of the new version of the protocol (higher gain of the "raw" version compared to the "active" version by up to 6.15 pp).

Inlining. In the inlining technique, the current trends may also influence the decision regarding its further use (Table 2).

Table 2. Comparison of Page Load Time for **inlining** scenarios.

Inlining		HTTP/1.1			HTTP/2.0					
		raw	active		raw		active		Server push	
		PLT [ms]	PLT [ms]	gain [%]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]
DSL	Chrome PC	148.0	149.0	−0.68%	139.0	6.08%	147.0	0.68%	134.0	9.46%
	Firefox	238.5	173.0	27.46%	171.5	28.09%	154.5	35.22%	175.0	26.62%
	Chrome mobile	287.5	246.5	14.26%	272.0	5.39%	244.0	15.13%	268.0	6.78%
3G	Chrome PC	449.0	426.0	5.12%	645.0	−43.65%	428.5	4.57%	624.0	−38.98%
	Firefox	1889.0	1233.0	34.73%	1497.0	20.75%	1427.5	24.43%	1444.0	23.56%
	Chrome mobile	850.0	509.5	40.06%	850.0	0.00%	508.5	40.18%	683.0	19.65%

If 3G users make up a significant share of the website, it is highly recommended to adhere to the technique (the highest PLT gain for the "active" version, the biggest difference visible for Chrome PC: −32.65% gain for the "raw" version, 4.57% for the "active" version, −38.98% for the "server push" version).

With increased participation of users with permanent Internet access and using Google Chrome browser, consider switching to the server push technique (increase by 8.78 pp in relation to the "active" version and 3.38 pp in relation to the "raw" version).

Domain Sharding. The domain sharding technique only worked in the 3G user scenario, equipped with the Google Chrome browser (40.07% PLT gain for the "active" version, 14.34% gain for the "raw" version) (Table 3).

Table 3. Comparison of Page Load Time for **sharding** scenarios.

Domain sharding		HTTP/1.1			HTTP/2.0			
		raw	active		raw		active	
		PLT [ms]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]
DSL	Chrome PC	251.0	257.0	−2.39%	215.0	14.34%	255.0	−1.59%
	Firefox	345.5	406.5	−17.66%	347.5	−0.58%	505.0	−46.16%
3G	Chrome PC	1078.0	478.0	55.66%	1047.5	2.83%	646.0	40.07%
	Firefox	2118.0	1830.0	13.60%	1703.0	19.59%	2256.0	−6.52%

If user participation is largely composed of this type of profile, applying the technique may be considered. Otherwise, it is recommended to abandon the

technique for the use of a single domain (gain decreases of up to 45.58 pp in relation to the "raw" version).

Minification. The situation with the use of minification is clear (Table 4). It is strongly recommended to continue using the technique to reduce the size of resources, which in each scenario yields a positive result in the context of PLT (gain increase of 79.52 pp compared to the "raw" version for the DSL profile and 143.97 pp in relation to the "raw" version for the 3G profile).

Table 4. Comparison of Page Load Time for **minification** scenarios.

Minification		HTTP/1.1			HTTP/2.0			
		raw	active		raw		active	
		PLT [ms]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]	PLT [ms]	Gain [%]
DSL	Chrome PC	7181.5	1727.5	75.95%	7464.0	−3.93%	1753.0	75.59%
	Firefox	7211.5	1766.5	75.50%	7531.0	−4.43%	1876.0	73.99%
	Chrome mobile	7581.0	2073.0	72.66%	7866.0	−3.76%	2014.5	73.43%
3G	Chrome PC	8411.0	1518.5	81.95%	15915.5	−89.22%	3806.0	54.75%
	Firefox	11121.0	3337.0	69.99%	17922.0	−61.15%	4970.0	55.31%
	Chrome mobile	9246.0	1902.0	79.43%	16496.0	−78.41%	4074.5	55.93%

5 Summary and Directions of Further Work

Properly designed, isolated simulation environment allowed to perform precise performance measurements of the websites examined in terms of the speed of loading the site during the first visit. Based on a precise analysis of the cleaned measurement data, it was possible to draw conclusion concerning the benefits of using classic techniques for designing efficient websites, including the HTTP/2 protocol. Based on these conclusions, a number of proposals were formulated regarding changes in the approach to already designed sites in terms of efficient charging using the old version of the protocol.

It turns out that not all techniques used in the context of the HTTP/1.1 protocol are beneficial after switching to the new version of the protocol. However, the recipients of the site should also be taken into account. Depending on the browser used or the available internet connection, the performance result may be different, as shown by many examples of specific visit scenarios.

For the concatenation technique, the only scenario convincing one to stick to it was the DSL scenario using the Firefox browser. In all other cases, the results were worse compared to the version without using the technique.

In the resource inlining scenarios using the 3G network proved to be beneficial. In the case of DSL networks, the use of the Google Chrome browser in the desktop version showed better results when using the alternative technique for resource inlining, i.e. server push.

The domain sharding technique proved to be useful only for a 3G user profile using the Google Chrome browser. Any other scenario indicated resignation from the technique as a beneficial step to increase efficiency.

The minification confirmed its "evergreen solution" title. The use of the technique did not lose its meaning after changing the protocol version to HTTP/2.

To sum up, when planning changes to the site design motivated by the use of the new version of the HTTP protocol, it is necessary to consider the nature of the site and who its recipient is, as it can have a significant impact on the final decision.

In the course of designing the simulation environment, the adoption of findings regarding the method of measuring size, the characteristics of visits, the interpretation of data, a number of aspects were found that can be subjected to improvements or changes. After their application, it shall be possible to conduct even more valuable substantive research and to draw new conclusions from them.

One of the aspects is to move the server environment to the cloud to examine real conditions. It would be necessary to conduct a larger number of measurements in the longer term, due to the fluctuations associated with the real network environment.

Another aspect is to examine a greater number of classic techniques that reveal their properties and benefits only in real network conditions.

The final suggestion for further work would be to further broaden the scenarios with new client profiles (browsers, devices) as well as network profiles (for example 4G, LTE and the like).

References

1. Grigorik, I.: High Performance Browser Networking. What Every Web Developer Should Know About Networking and Web Performance. O'Reilly Media, Sebastopol (2013)
2. Grigorik, I.: HTTP/2: A New Excerpt from High Performance Browser Networking. O'Reilly Media, Sebastopol (2015)
3. Liu, Y., Liu, X., Huang, G.: Can HTTP/2 really help web performance on smartphones? In: 2016 IEEE International Conference on Services Computing (2016)
4. Corbel, R., Stephan, E., Omnes, N.: HTTP/1.1 pipelining vs HTTP2 in-the-clear: performance comparison. In: 3th International Conference on New Technologies for Distributed Systems (2016)
5. de Saxcé, H., Oprescu, I., Chen, Y.: Is HTTP/2 Really Faster Than HTTP/1.1? In: 18th IEEE Global Internet Symposium (2015)
6. Varvello, M., Schomp, K., Naylor, D., Blackburn, J., Finamore, A., Papagiannaki, K.: Is the Web HTTP/2 Yet? In: 17th International Conference Of Passive and Active Measurement, PAM 2016 (2016)
7. Zarifis, K., Holland, M., Jain, M., Katz-Bassett, E., Govindan, R.: Modeling HTTP/2 speed from HTTP/1 traces. In: 17th International Conferenceof Passive and Active Measurement, PAM 2016 (2016)

8. Stępniak, W., Nowak, Z.: Performance analysis of SPA web systems. In: Borzemski, L., Grzech, A., Świątek, J., Wilimowska, Z. (eds.) Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016 – Part I. AISC, vol. 521, pp. 235–247. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-46583-8_19
9. Rosen, S., Han, B., Hao, S., Morley Mao, Z., Qian, F.: Push or request: an investigation of HTTP/2 server push for improving mobile performance. In: The 26th International Conference (2017)
10. Kim, H., Lee, J., Park, I., Kim, H., Yi, D., Hur, T.: The upcoming new standard HTTP/2 and its impact on multi-domain websites. In: Asia-Pacific Network Operations and Management Symposium (2015)
11. Seemakhupt, K., Piromsopa, K.: When should we use HTTP2 multiplexed stream? In: 13th International Joint Conference on Computer Science and Software Engineering (2016)
12. Prokopiuk, J., Nowak, Z.: The influence of HTTP/2 on user-perceived Web application performance. Studia informatica, vol. 38, no. 3(132). Silesian University of Technology Press, Gliwice (2017)
13. Souders, S.: High Performance Web Sites. Essential Knowledge for Front-End Engineers. O'Reilly Media, Sebastopol (2007)
14. Ndegwa, A.: What is Page Load Time? MaxCDN One (2016). https://www.maxcdn.com/one/visual-glossary/page-load-time/
15. Dutton, S.: Measuring page load speed with navigation timing, blog HTML5 rocks (2011). https://www.html5rocks.com/en/tutorials/webperformance/basics/
16. Grigorik, I.: Measuring network performance with resource timing API, blog Google developers (2013). https://developers.googleblog.com/2013/12/measuring-network-performance-with.html
17. Trosien, O.: HTTP2 w/ Spring Boot+Jetty. GitHub (2017). https://github.com/otrosien/demo-http2
18. Rizzo, L.: The Dummynet Project. University of Pisa (2015). http://info.iet.unipi.it/~luigi/dummynet/